PCT WORLD INTELLECTUAL PROPI International B

INTERNATIONAL APPLICATION PUBLISHED UNDEF

WO 9608773A2

(51) International Patent Classification 6 : G06F 13/28 | A2

(11) Ir

(43) International Publication Date: 21 March 1996 (21.03.96)

(21) International Application Number: PCT/US95/11692

(22) International Filing Date: 15 September 1995 (15.09.95)

(30) Priority Data:
08/307,965 16 September 1994 (16.09.94) US

(71) Applicant: CIRRUS LOGIC, INC. [US/US]; B5-906, 3100 West Warren Avenue, Fremont, CA 94538-6423 (US).

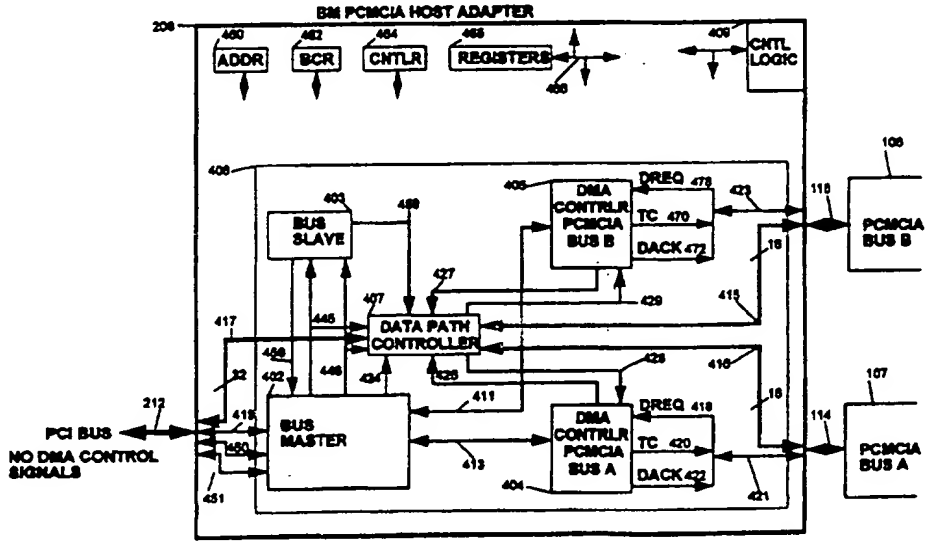
(72) Inventors: BEZZANT, Daniel, G.; 1884 Paseo del Cajon, Pleasanton, CA 94566 (US). SMITH, Stephen, A.; 760 Homer Avenue, Palo Alto, CA 94301 (US). NOOKALA, Narasimha, R.; 3604 Sydney Drive, San Jose, CA 95132 (US). NARAYANAN, Puducode, S.; 243 Buena Vista Avenue #805, Sunnyvale, CA 94086 (US). DIKSHIT, Ashutosh, S.; 100 N. Whisman Road #290, Mountain View, CA 94043 (US).

(74) Agents: CHURCH, Shirley, L. et al.; Cirrus Logic, Inc., B5-906, Legal/IP, 3100 West Warren Avenue, Fremont, CA 94538-6423 (US).

(81) Designated States: CN, JP, KR, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: PCMCIA DMA DATA BUS MASTERING



(57) Abstract

A PCMCIA host adapter includes the capability to master a non-DMA system bus and control a DMA data transfer between a DMA capable peripheral and the internal system memory. A peripheral can be coupled to the system through a PCMCIA card plugged into a PCMCIA expansion slot. A DMA controller coupled to the PCMCIA expansion slot through a PCMCIA bus controls a DMA transfer between the internal system memory and the peripheral. A bus master disables the CPU and takes control of the system bus during a DMA data transfer. In an alternative embodiment, the PCMCIA host adapter can be used with either a system having a system bus with DMA capability or with a system having a system bus without DMA capability. In this alternate embodiment if the system bus has DMA capability, the PCMCIA host adapter effectively passes the DMA signals between the peripheral and the system bus. If the system bus does not have DMA capability then the DMA controller and the bus master work to disable the CPU and take control of the system bus during a DMA data transfer. The DMA controller then controls the transfer of data between the peripheral and the internal system memory.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria
AU	Australia
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CS	Czechoslovakia
CZ	Czech Republic
DE	Germany
DK	Denmark
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgystan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakhstan
LI	Liechtenstein
LK	Sri Lanka
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SI	Slovenia
SK	Slovakia
SN	Senegal
TD	Chad
TG	Togo
TJ	Tajikistan
TT	Trinidad and Tobago
UA	Ukraine
US	United States of America
UZ	Uzbekistan
VN	Viet Nam

PCMCIA DMA DATA BUS MASTERING

FIELD OF THE INVENTION

The present invention relates to the field of performing DMA transfers between a computer and peripheral devices which are connected to that computer through PCMCIA cards. More particularly, the present invention relates to the field of performing DMA transfers between a computer and peripheral devices where the system bus within the computer does not have DMA capability.

BACKGROUND OF THE INVENTION

DMA DATA TRANSFER

During a computer's operation it is often necessary for data to be transferred between an external mass storage device, such as a magnetic disk or a magnetic tape, and the internal memory within the computer system. The speed of this transfer is limited by the speed of the microprocessor within the computer. Using a direct memory access (DMA) technique, it is possible to bypass the microprocessor during such a transfer and allow the peripheral device to directly transfer data to or from the internal system memory, thus improving the speed of the transfer and correspondingly the efficiency of the system.

During a DMA transfer, the microprocessor gives up control of the system bus and a DMA controller takes control of the system bus to manage the transfer directly between the peripheral device and the internal system memory. The

microprocessor is then free to perform other calculations and operations while the DMA transfer is taking place as long as those other calculations and operations do not require the system bus.

Using a DMA technique, the transfer of data can be made for an entire block of memory words in a single stream of data or the transfer can be made by cycle stealing or transferring one word at a time in between microprocessor instruction executions. During cycle stealing, the microprocessor will delay its operation for one memory cycle to allow the direct memory transfer controller to steal one memory cycle. The microprocessor will then continue its operation and will allow the DMA data transfer to take place, on the system busses, between the execution of the microprocessor's instructions.

The DMA controller is an interface between the microprocessor, the peripherals and the internal system memory and therefore requires circuits necessary for communication with the microprocessor. In addition, the DMA controller includes an address register, a byte count register, a control register and a set of address lines. The address register and the address lines are used for direct communication with the internal system memory. The byte count register specifies the number of bytes remaining to be transferred by the DMA controller. The control register specifies the direction of the data transfer. The data transfer is usually done directly between the specific peripheral device and the internal system memory, utilizing the data bus, under the supervision of the DMA controller.

A bus request signal line and a bus granted signal line are coupled between the microprocessor and the DMA controller. The bus request signal line, sometimes referred to as a hold command, is used by the DMA controller to request that the microprocessor disable its busses and allow the DMA controller to utilize them. The bus granted signal line, sometimes referred to as a hold acknowledge, is used to notify the DMA controller that the microprocessor has given up control of the system busses.

When it is necessary to transfer data from a peripheral device to the internal system memory the bus request signal line is raised by the DMA controller to a logical high voltage level. The microprocessor then completes the execution of the present instruction and places its busses, including the read and write signal lines, into a high-impedance state and then raises the bus granted signal line to a logical high voltage level. As soon as the bus granted signal line is raised to a logical high voltage level, the DMA controller can take control of the system bus to communicate directly with the internal system memory. As long as the bus granted signal line is at a logical high voltage level the microprocessor does not have control of the system busses. To signal that the transfer of data is complete, the DMA controller will lower the bus request signal line to a logical low voltage level. The microprocessor will return to its normal operation after the bus request signal line is lowered to a logical low voltage level by lowering the bus granted signal line to a logical low voltage level and enabling its busses. The DMA controller also communicates with an external peripheral

device through request and acknowledge signal lines in order to initiate a DMA transfer of data.

As stated above, the DMA controller includes an address register, a byte count register and a control register. The address register is used to specify the desired location of the data transfer within the internal system memory. The address register is incremented after each DMA byte transfer. The byte count register stores the number of bytes remaining to be transferred to or from the internal system memory. This register is decremented after each DMA byte transfer and is internally tested for zero. Once the byte count is equal to zero, the DMA controller lowers the bus request signal line to a logical low voltage level, signalling that the transfer of data is complete. Alternatively, the byte count register can store the number of bytes that have already been transferred. The byte count register will then be initialized to zero and incremented after each DMA byte transfer. In this case, the byte count register is internally compared against the number of total bytes of data to be transferred. The control register specifies the direction of the data transfer, whether it is into (write) or out of (read) the internal system memory. The microprocessor can read from or write into the three registers within the DMA controller.

A data transfer using DMA is first initialized by the microprocessor. The initialization is essentially a program consisting of instructions that include loading the three registers within the DMA controller with the specific information regarding the DMA transfer and sending a control signal from the microprocessor to the DMA controller

signalling that the DMA transfer should begin. During initialization, the address register is loaded with the address within the internal system memory where the transfer is to take place, the byte count register is loaded with the number of bytes to be transferred, the control register is loaded with the information regarding the direction of the data transfer and the control signal is sent to the DMA controller. After this initialization, the DMA controller starts and continues to transfer data between the internal system memory and the specific peripheral until an entire block of data is transferred.

A peripheral device can also initiate a DMA transfer by sending the DMA controller a DMA request signal. When the DMA controller receives a DMA request signal from a peripheral device, it activates its bus request signal line, informing the microprocessor that the DMA controller wishes to transfer data. As explained above the microprocessor then completes the execution of the present instruction and places its busses, including the read and write lines, into a high-impedance state. Once the DMA controller has control of the bus system it then places the current value of its address register onto the address bus, initiates the read or write signal and sends a DMA acknowledge signal to the peripheral device. The peripheral device then puts a byte of data on the data bus for a write operation or receives a byte of data from the data bus for a read operation. Thus, the DMA controller controls the read or write operation and supplies the address for the internal system memory. The peripheral device can then communicate with the internal

system memory through the data bus for direct transfer between the two units while the microprocessor is disabled.

For each byte of data that is transferred, the DMA controller increments its address register and decrements its byte count register. If the value stored in the byte count register is not at zero, the DMA controller checks the request line from the peripheral. For a high-speed peripheral, the request line is activated as soon as the previous transfer is completed, a second transfer is then initiated and the process continues until the entire block of data is transferred. If the speed of the peripheral is slower, the request signal from the peripheral may come somewhat later. In this case, the DMA controller removes its bus request and allows the microprocessor to continue to execute its program. When the peripheral catches up and requests a transfer, the DMA controller will request the busses from the microprocessor again.

When the byte count register reaches zero, the DMA controller stops any further transfer of data and removes its bus request. The DMA controller also uses an interrupt request to inform the microprocessor that the data transfer has been terminated. When the microprocessor responds to the DMA interrupt, it reads the contents of the byte count register. The zero value of this register indicates that all of the bytes of data were successfully transferred. The microprocessor can read the value stored in the byte count register at any other time as well, in order to check the number of bytes of data that have already been transferred.

SYSTEM BUSSES

System busses are used to allow circuits within a computer system to communicate with other circuits, such as between peripherals connected to the system and the internal system memory, as described above. A system bus consists of multiple signal lines, including address lines, data lines, Read/Write or handshake lines and other control signal lines. The size of a system bus is dependent on the capabilities of the CPU which is used to control the system bus.

Computer manufacturers have typically designed the internal computer system bus structure around the microprocessor that they support. Because each manufacturer supports its own microprocessor, numerous bus structures have been developed and become standards. IBM developed and uses the ISA bus standard for the IBM PC system bus structure, Apple Computer has developed and uses the NUBUS standard for the system bus architecture of some Macintosh computers, Intel has developed and uses the PCI bus standard and VESA (Video Electronics Standard Association) has developed and uses the Local Bus standard (VL Bus). There are many other bus structures used within various electronic equipment, but most of these system busses have not become popular standards because either the equipment is not used widely or the system bus is proprietary and is not disclosed. Not all of these bus structures have the capability to support DMA data transfer.

Of interest herein are the computer system busses developed around Intel's family of microprocessors, including the ISA bus and the PCI bus. However, it will become clear that the present invention can be applied to other bus structures as well.

The ISA bus standard is used in almost every IBM PC or PC clone and accordingly has become a very popular bus standard. The ISA bus standard was created by IBM and has a 16 bit wide data path and a 24 bit wide address bus. The ISA bus is asynchronous and originally operated with system clock rates of 8MHz. Because the ISA bus standard has remained fixed, it has become limited for use in conjunction with modern high speed microprocessors. However, the ISA bus does have the capability to support DMA data transfers.

The PCI bus standard was created by Intel Corporation for use in notebook and laptop computers. The PCI bus standard has power saving features as well as a faster bandwidth rate than the ISA bus standard. The PCI bus standard also utilizes a 16 bit wide data bus and a 24 bit wide address bus. However, the PCI bus does not have the capability to support DMA data transfers.

The Personal Computer Memory Card International Association (PCMCIA) is an association which sets standards and specifications by which a peripheral communicates to a host adapter or CPU through an interface. A PCMCIA card is used as this standard interface and allows peripheral devices including a modem card, a network card, a sound card, a floppy disk drive and a hard disk drive to be coupled to the system computer through a PCMCIA card. This PCMCIA card is plugged into a PCMCIA expansion slot which is

coupled to a PCMCIA adapter within the computer system. The PCMCIA standard enables memory and I/O devices to be inserted as exchangeable peripherals into personal and handheld computers.

A block diagram of a computer system which has PCMCIA capability is illustrated in Figure 1A. A motherboard 102 is coupled to a CPU 110, a PCMCIA host adapter 108, a DMA controller 111, an internal system memory 113, a floppy disk drive 120, a hard disk drive 121 and a graphics controller 122. The system busses 112 couple all of the above listed circuits to the CPU 110. The graphics controller 122 is also coupled to the display 101 and the system busses 112 are also coupled to a keyboard or other I/O device 123. The CPU 110 enables the DMA controller 111 to control the system busses 112 for a DMA transfer between a peripheral and the internal system memory 113. The PCMCIA host adapter 108 is also coupled to the two PCMCIA expansion slots 106 by the PCMCIA bus signal lines 114. As peripherals are added to the system, a PCMCIA card 104 or 105 is plugged into one of the PCMCIA expansion slots 106. When a DMA transfer is requested from a peripheral coupled to the PCMCIA host adapter 108, the PCMCIA adapter 108 requests the DMA controller 111 to initialize and execute the DMA transfer.

At the present time, DMA capability is not required by the PCMCIA specification and standards. However, there are peripherals which can be connected to the system computer through the PCMCIA expansion slots 106 and can be operated using a DMA transfer protocol. There are also system busses, such as the ISA bus, which also support DMA transfers. When a DMA transfer is to take place between a

PCMCIA peripheral which has DMA capability and a system using a system bus also having DMA capability, the PCMCIA host adapters 108 of the prior art will effectively pass through the necessary DMA signals from the system bus 112 to the PCMCIA bus 114 as illustrated in Figure 1B.

One of the PCMCIA cards 104 is coupled to a peripheral which does not have DMA capability. The other PCMCIA card 105 is coupled to a peripheral which does have DMA capability. When a PCMCIA card 104 or 105 is plugged into the PCMCIA expansion slot, the PCMCIA host adapter 108 first interrogates the PCMCIA card 104 or 105 to determine whether or not it has DMA capability. As long as that card remains plugged into the expansion slot 106, the PCMCIA host adapter 108 will know that it does or does not have DMA capability.

What is needed is a PCMCIA Host Adapter which can be coupled to a system bus which does not support DMA transfers and will take over the system bus during data transfers between the system memory and the peripherals plugged into the PCMCIA slots and effectively control a DMA data transfer between the peripheral and the system bus.

SUMMARY OF THE INVENTION

A PCMCIA host adapter includes the capability to master a non-DMA system bus and control a DMA data transfer between a DMA capable peripheral and the internal system memory. A peripheral can be coupled to the system through a PCMCIA card plugged into a PCMCIA expansion slot. The PCMCIA host adapter further includes a DMA controller coupled to the PCMCIA expansion slots through a PCMCIA bus to control a DMA transfer between the internal system memory and the

peripheral. A bus master circuit coupled to the DMA controller disables the CPU and takes control of the system bus during a DMA data transfer. In an alternative embodiment, the PCMCIA host adapter can be used with either a system having a system bus with DMA capability or with a system having a system bus without DMA capability. In this alternate embodiment if the system bus has DMA capability, the PCMCIA host adapter effectively passes the DMA signals between the peripheral and the system bus. If the system bus does not have DMA capability then the DMA controller and the bus master work to disable the CPU and take control of the system bus during a DMA data transfer. The DMA controller then controls the transfer of data between the peripheral and the internal system memory.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1A illustrates a block diagram of a prior art system which has PCMCIA capability.

Figure 1B illustrates a PCMCIA Host Adapter according to the prior art which effectively passes through DMA transfer signals.

Figure 2 illustrates a block diagram of a system according to the present invention incorporating a PCMCIA Host Adapter which includes a DMA controller.

Figure 3 illustrates a PCMCIA Host Adapter according to the present invention which is coupled to a DMA capable system bus and to a peripheral having DMA capability.

Figure 4 illustrates a block diagram of a PCMCIA Host Adapter according to the present invention for controlling two PCMCIA expansion slots.

Figure 5 illustrates a block diagram of a bus master logic circuit and a DMA controller logic circuit according to the present invention.

Figure 6 illustrates a block diagram of a data path controller logic circuit according to the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A block diagram of a computer system according to the preferred embodiment of the present invention is illustrated in Figure 2. This system includes a PCMCIA host adapter 208 having bus mastering capability, which is coupled to the CPU 110, to a bus arbiter 220 and to the system memory 113 by a system bus 212 which does not have DMA capability, such as the PCI bus. The PCMCIA host adapter 208 includes a DMA controller which allows DMA transfers to take place on the non-DMA system bus, between DMA capable peripherals coupled through the PCMCIA cards 104 and 105 and the internal system memory 113. The bus arbiter 220 determines which circuit within the system gets control of the system busses 212.

In an alternate embodiment of the present invention, a single PCMCIA host adapter 208 could be used with both system busses having DMA capability and system busses which do not have DMA capability. A control bit within this embodiment of the PCMCIA host adapter 208 can be used to specify whether or not the system bus 112 has DMA capability. If the system bus 112 does have DMA capability, then the control bit is set and the PCMCIA host adapter 208 will couple the system bus 112 to the PCMCIA bus 114 and effectively pass the DMA signals through the PCMCIA host adapter 208 as illustrated in Figure 3. If the system bus

112 does not have DMA capability, then the control bit is reset and the PCMCIA host adapter 208 will couple the DMA signals to the DMA controller 404 which will then control any DMA transfers as discussed with reference to and illustrated in Figure 4. This single PCMCIA host adapter 208 having dual bus compatibility could then be used in either a system having an ISA bus or a system having a PCI bus, for example.

A block diagram of a PCMCIA Host Adapter 208 according to the preferred embodiment of the present invention is illustrated in Figure 4. The PCMCIA Host Adapter 208 is coupled to a system bus 212 which does not support DMA data transfers and to the PCMCIA bus for slot A 114 and the PCMCIA bus for slot B 115. The PCMCIA host adapter 208 includes a DMA control engine 406 which controls the transfer of data to and from each of the PCMCIA expansion slots 106, control logic 409 for controlling the operation of the PCMCIA host adapter 208 and the internal registers 465, including an address register 460, a byte count register 462 and a control register 464.

The DMA Controller 404 for controlling the DMA transfers across the PCMCIA bus A 114 is coupled to the PCMCIA bus 114 by a DMA request signal line 418, a DMA acknowledge signal line 422 and a terminal count signal line 420. The DMA controller 404 is also coupled to the bus master 402 by the signal lines 413 and to the data path controller 407 by the signal lines 426 and 428. The bus master 402 is also coupled to the system bus 212, to the data path controller 407 by the signal lines 424 and 446 and to the bus slave circuit 403 by the signal lines 446 and

458. The bus slave circuit 403 is also coupled to the data path controller 407 by the signal line 459.

The DMA controller 405 for controlling DMA transfers across the PCMCIA bus B 115 is coupled to the PCMCIA bus 115 by a DMA request signal line 478, a DMA acknowledge signal line 472 and a terminal count signal line 470. The DMA controller 405 is also coupled to the bus master 402 by the signal line 411 and to the data path controller 407 by the signal lines 427 and 429. The DMA controller 404 and the DMA controller 405 are designed to operate identically when data is being sent to or received from the peripheral coupled to their respective PCMCIA bus. Therefore, the operation of only the DMA controller 404 will be described in detail below.

The PCMCIA host adapter 208 includes an address register 460, a byte count register 462 and a control register 464 as described above. When the system is required to execute a data transfer, the address register 460 is loaded with the desired location of the data transfer within the internal system memory 113, the byte count register 462 is loaded with the number of bytes to be transferred and the control register 464 is loaded with the direction of the data transfer, either into the internal system memory 113 or out of the internal system memory 113. After loading the respective values into the address register 460, the byte count register 462 and the control register 464, the DMA controller 404 signals to the bus master 402 that it should request control of the system bus 212. The bus master 402 then negotiates with the bus arbiter 220 for control of the system bus 212. The bus

master 402 sends a request signal to the bus arbiter 220 over the bus request signal line 450. The bus arbiter 220 allows the CPU 110 to finish its current instruction before it allows the bus master 402 to take control of the system bus 212. The bus arbiter 220 then signals to the bus master 402 that it can now control the system bus 212, over the bus granted signal line 451. Once the bus master 402 has control of the system bus 212, the DMA controller 404 then controls the DMA transfer between the internal system memory 113 and the respective peripheral coupled through a PCMCIA card 104 or 105.

When the DMA data transfer is complete, the DMA controller 404 then notifies the bus master 402 that it no longer requires control of the system bus 212. The bus master 402 will then relinquish control of the system bus 212 to the bus arbiter 220 by taking away the signal on the request signal line 450. If for some reason, during the DMA transfer, the CPU must take back control of the system bus 212, the bus arbiter 220 will change the signal on the bus granted signal line 451, signalling to the bus master 402 that it no longer controls the system bus 212.

The DMA controller 404, the bus master 402 and the bus slave circuit 403 work together with the data path controller to control the direction of data flow between the system bus 212 and the PCMCIA bus 114. The bus slave circuit 403 provides the interface to the system bus 212 when the CPU 110 or another circuit within the system has control of the system bus 212. The bus master 402 provides the interface to the system bus 212 when the PCMCIA host adapter 208 has control of the system bus 212 and is

controlling a DMA data transfer. The data path controller 407 controls the direction of the data flow through the PCMCIA host adapter 208, either from the PCMCIA expansion slots 106 to the system bus 212 or from the system bus 212 to the PCMCIA expansion slots 106.

A more detailed block diagram of the bus master 402 and the DMA controllers 404 and 405 of the preferred embodiment is illustrated in Figure 5. Within the bus master 402, the master state machine 518 is coupled to the system bus handshake control circuit 502 by the bi-directional signal line 530, to the master data transfer controller 504 by the bi-directional signal line 532, to the bus mastering controller 516 by the bi-directional signal line 552 and to the system bus direction control circuit 520 by the signal line 554. The system bus direction control circuit 520 is coupled to the system bus master/slave signal multiplexer 522 by the signal line 556. The system bus master/slave signal multiplexer 522 is also coupled to the system bus handshake control circuit 502 by the signal line 558.

The system bus master/slave signal multiplexer 522 is also coupled to receive the system bus control signals 419 including the wait signal, the read and write signals and the address signals from the system bus 212. The system bus master/slave signal multiplexer 522 is also coupled to the bus slave circuit 403 by the bus slave control signal line 458, which notifies the system bus master/slave signal multiplexer 522 that the bus slave circuit 403 is controlling the interface between the system bus 212 and the PCMCIA host adapter 208. The bus mastering controller 516 is also coupled to the bus arbiter 220 by the bus request

signal line 450 and the bus granted signal line 451. The master state machine 518 is also coupled to the data path controller 407 by the data path control signal line 424.

Within the DMA controller 404, the DMA transfer control state machine 508 is coupled to the PCI bus address generator 506 by the signal line 538, to the PCMCIA DMA cycle generator 510 by the bi-directional bus 544 and to the bytes transfer counter 512 by the signal lines 540 and 542. The DMA transfer control state machine 508 is also coupled to the data path controller 407 by the data path control signal line 426 and the data prefetch and FIFO status signal line 428, to the master data transfer controller 504 by the bi-directional signal line 534 and to the bus mastering controller 516 by the bi-directional signal line 548. The word/double word conversion control circuit 514 is also coupled to the master state machine 518, the master data transfer controller 504 and the data path controller 407 by the signal line 446. The system bus address generator 506 is also coupled to the system bus master/slave signal multiplexer 522 by the signal line 536. The PCMCIA DMA cycle generator 510 is also coupled to the PCMCIA control signal lines 421 for sending data to and receiving data from the peripheral plugged into the PCMCIA expansion slot A.

Within the DMA controller 405, the DMA transfer control state machine 508 is coupled to the PCI bus address generator 506 by the signal line 538, to the PCMCIA DMA cycle generator 510 by the bi-directional bus 544 and to the bytes transfer counter 512 by the signal lines 540 and 542. The DMA transfer control state machine 508 is also coupled to the data path controller 407 by the data path control

signal line 427 and the data prefetch and FIFO status signal line 429, to the master data transfer controller 504 by the bi-directional signal line 533 and to the bus mastering controller 516 by the bi-directional signal line 547. The word/double word conversion control circuit 514 is also coupled to the master state machine 518, the master data transfer controller 504 and the data path controller 407 by the signal line 445. The system bus address generator 506 is also coupled to the system bus master/slave signal multiplexer 522 by the signal line 535. The PCMCIA DMA cycle generator 510 is also coupled to the PCMCIA control signal lines 423 for sending data to and receiving data from the peripheral plugged into the PCMCIA expansion slot B.

A detailed block diagram of the data path controller 407 is illustrated in Figure 6. The flow control circuit 622 controls the direction of the data flow between the system bus and the PCMCIA busses A and B. Because the PCI bus 417 is a thirty-two bit bus and the PCMCIA busses 415 and 416 are sixteen bit busses, the data path controller 407 must also pack two sixteen bit words from the PCMCIA bus 416 onto the PCI bus 417 or unpack a double word from the PCI bus 417 into two sixteen bit words and send them over the PCMCIA bus 416 one at a time, in the right order. The data path controller 407 also includes two sixteen bit read/write FIFOs 606 and 607 for prefetching data during a read or write instruction. The PCI bus 417 is coupled to the system bus 212. The PCMCIA bus A is coupled to the bus 114. The PCMCIA bus B is coupled to the bus 115.

The flow control circuit 622 is coupled to the master state machine 518 by the data path control signal line 424,

to the word/double word conversion control circuit 514 by the signal line 446, to the DMA transfer control state machine 508 within the DMA controller 404 by the data path control signal line 426 and the data prefetch and FIFO status signal line 428 and to the DMA transfer control state machine 508 within the DMA controller 405 by the data path control signal line 427 and the data prefetch and FIFO status signal line 429. The flow control circuit 622 is also coupled to the FIFO control circuits 604 and 605 by the bi-directional signal line 626, to the data latch 616 by the control signal line 638, to the data latch 615 by the control signal line 636, to the data latches 608 and 609 by the control signal line 632, to the tri-state buffer 611 by the control signal line 633, to the tri-state buffer 612 by the control signal line 635 and to the tri-state buffers 613 and 614 by the control signal line 634.

The PCMCIA sixteen bit bus and control signals 416 are coupled to the read/write FIFO 606. The PCMCIA sixteen bit bus and control signals 415 are coupled to the read/write FIFO 607. The FIFO control circuit 604 is coupled to the read/write FIFO 606 by the control signal line 624 and the FIFO control circuit 605 is coupled to the read/write FIFO 605 by the control signal line 624. The FIFO control circuits 604 and 605 are also coupled to the bus slave circuit 403 by the control signal line 459. The read/write FIFOs 606 and 607 are also coupled to the inputs of the latches 615 and 616 and to the outputs of the tri-state buffers 611 and 612 by the sixteen bit bus 645. The output of the sixteen bit latch 615 is coupled to the input of the tri-state buffer 613 by the bus 642. The output of the

sixteen bit latch 616 is coupled to the input of the tri-state buffer 614 by the bus 643.

The thirty-two bit PCI bus 417 is split into a sixteen bit upperword and a sixteen bit lower word. The sixteen bit upper word of the PCI bus 417 is coupled to the input of the latch 608 and to the output of the tri-state buffer 613. The sixteen bit lower word of the PCI bus 417 is coupled to the input of the latch 609 and to the output of the tri-state buffer 614. The output of the sixteen bit latch 608 is coupled to the input of the tri-state buffer 611 by the bus 640. The output of the sixteen bit latch 609 is coupled to the input of the tri-state buffer 612 by the bus 641.

When the PCMCIA host adapter 208 is not performing a DMA transfer and therefore does not have control of the system bus 212, the bus slave circuit 403 and the data path controller 407 work together to control the flow of data through the PCMCIA host adapter 208. When the PCMCIA host adapter 208 is performing a DMA transfer and therefore does have control of the system bus 212, the bus master 402 and all of its internal circuits, the respective one of the DMA controllers 404 or 405 and all of its internal circuits and the data path controller work to control the DMA transfer of data through the PCMCIA host adapter 208.

A peripheral device having DMA capability can also initiate a DMA transfer through the PCMCIA host adapter 208 of the present invention. The peripheral device plugged into the PCMCIA expansion slot A 107 first sends a DMA request signal on the signal line DREQ 418 to the DMA controller 404. When the DMA controller 404 receives a DMA request signal from the peripheral device, it notifies the

bus master 402 that it wishes to perform a DMA transfer and will need control of the system bus 212. The bus master 402 then sends a bus request signal on the bus request signal line 450 to the bus arbiter 220. The bus arbiter 220 then waits until the CPU 110 completes its present instruction, disables the CPU 110 and then notifies the bus master 402 on the bus granted signal line 451 that it can now have control of the system bus 212. Once the bus master 402 has control of the system bus 412, it then notifies the DMA controller 404 that it has control of the system bus 412. The DMA transfer is then initialized by loading the respective values in the address register 460, the byte count register 462 and the control register 464. In the preferred embodiment of the present invention the byte count register 462 is initialized to zero and compared against the total number of bytes to be transferred during the DMA transfer. The current value stored within the address register is then placed onto the address bus, and the DMA controller 404 then initiates the read or write signal and sends a DMA acknowledge signal to the peripheral device on the signal line DACK 422. The data path flow control circuit 622 is also notified from which direction the data transfer will be originating. If the data is to flow from the PCMCIA bus 416 to the PCI bus 417, the data flow control circuit 622 controls the data flow through the latches 615 and 616 and the tri-state buffers 613 and 614, combining two sixteen bit words into a thirty-two bit double word. If the data is to flow from the PCI bus 417 to the PCMCIA bus 416, the data flow control circuit 622 controls the data flow through the latches 608 and 609 and the tri-state buffers 611 and 612,

separating the thirty-two bit double word into two sixteen bit words. The word/double word conversion control circuit 514 controls the packing or unpacking of the data between the thirty-two bit system bus 212 and the sixteen bit PCMCIA bus 114. The FIFOs 606 and 607 are used to manage the data flow through the data path controller 407.

The peripheral device will send data to or receive data from the FIFO 606. Thus, the DMA controller controls the read or write operation and supplies the address for the internal system memory 113. The peripheral device can then communicate with the internal system memory 113 through the system bus 412 and the PCMCIA bus 416 for direct transfer between the two units while the CPU 110 is disabled.

For each byte of data that is transferred, the DMA controller 404 increments the value stored in its address register 424 and the value stored in its byte count register 426. The DMA controller 404 monitors the value stored in the byte count register 426. When the value stored in the byte count register 426 is equal to the total number of bytes to be transferred, indicating that the DMA transfer is complete, the DMA controller 404 raises the terminal count signal line TC 420 to a logical high voltage level signalling to the peripheral that the DMA transfer is complete. The terminal count signal line TC 420 is raised to a logical high voltage level in the preferred embodiment of the present invention during the last card read or write cycle of the DMA transfer process.

If a peripheral device which does not have DMA capability is coupled to the system through a PCMCIA card 104 then the PCMCIA host adapter 208 according to the

present invention will couple the system bus 412 to the PCMCIA bus 214, using the bus slave circuit 403 and the CPU 110 will control any data transfers which are required without using DMA. The PCMCIA host adapter 208 will then effectively pass the signals through the data path controller 407, in the direction of the transfer, controlled by the bus slave circuit 403.

While the system of the preferred embodiment has been designed to work with a PCI bus, the scope of the present invention should not be so limiting. It should be apparent to one skilled in the art that the principles of the present invention can be used with other system busses which do not support DMA transfer. It should also be apparent that the alternate embodiment of the present invention can also be used with system busses, other than the ISA bus, which do support DMA transfers. Other improvements and modifications which become apparent to persons of ordinary skill in the art only after reading this disclosure, the drawings and the appended claims are deemed within the spirit and scope of the present invention.

C L A I M S

We Claim:

1. A host adapter integrated circuit for use in a computer system having a CPU, an internal system memory, a system bus and one or more peripherals, the host adapter comprising:

a. a determining logic circuit coupled to the CPU, the system bus and each of the peripherals for determining whether the system bus has DMA capability;

b. a switching logic circuit coupled to the determining logic circuit, the system bus and each of the peripherals for coupling the peripherals directly to the system bus and effectively passing through DMA signals if the system bus does have DMA capability;

c. a controlling logic circuit coupled to the peripherals, the determining logic circuit and the internal system memory for controlling a DMA transfer if the system bus does not have DMA capability; and

d. a mastering logic circuit coupled to the controlling logic circuit, the determining logic circuit, the system bus and the CPU for disabling the CPU during a DMA transfer and taking control of the system bus if the system bus does not have DMA capability.

2. The host adapter as claimed in claim 1 wherein the peripherals are PCMCIA type peripherals.

3. The host adapter as claimed in claim 2 wherein there are two PCMCIA type peripherals.

4. The host adapter as claimed in claim 3 wherein the controlling circuit further comprises a first DMA controller circuit for controlling a first PCMCIA slot and a second DMA controller for controlling a second PCMCIA slot.

5. A host adapter integrated circuit for use in a computer system having a CPU, an internal system memory, a system bus and one or more peripherals, the host adapter comprising:

a. means for determining whether the system bus has DMA capability coupled to the CPU, the system bus and the peripherals;

b. means for coupling the peripherals directly to the system bus and effectively passing through DMA signals, if the system bus does have DMA capability, the means for coupling coupled to the means for determining, the peripherals and the system bus;

c. means for controlling a DMA transfer coupled to the peripherals, to the means for determining and to the internal system memory for controlling a DMA transfer if the system bus does not have DMA capability; and

d. means for mastering the system bus if the system bus does not have DMA capability, coupled to the means for controlling, to the means for determining, to the system bus and to the CPU for disabling the CPU during a DMA transfer and taking control of the system bus if the system bus does not have DMA capability.

6. The host adapter as claimed in claim 5 wherein the peripherals are PCMCIA type peripherals.

7. The host adapter as claimed in claim 6 wherein there are two PCMCIA type peripherals.

8. The host adapter as claimed in claim 7 wherein the means for controlling further comprises a first DMA controller circuit for controlling a first PCMCIA slot and a second DMA controller circuit for controlling a second PCMCIA slot.

9. A method for performing DMA transfers within a computer system having a CPU, an internal system memory, a system bus and one or more peripherals comprising the steps of:

a. determining whether the system bus has DMA capability;

b. coupling the peripherals directly to the system bus and effectively passing through the DMA signals if the system bus does have DMA capability;

c. mastering the system bus if the system bus does not have DMA capability; and

d. controlling a DMA transfer of data between a peripheral and the internal system memory.

10. The method as claimed in claim 9 wherein the peripherals are PCMCIA type peripherals.

11. The method as claimed in claim 10 wherein there are two PCMCIA type peripherals.

12. A PCMCIA host adapter integrated circuit for use in a computer system having a CPU, an internal system memory, a system bus and one or more PCMCIA slots, the PCMCIA host adapter comprising:

a. a determining logic circuit coupled to the CPU, the system bus and the PCMCIA slots for determining whether the system bus has DMA capability;

b. a switching logic circuit coupled to the determining logic circuit, the PCMCIA slots and the system bus for coupling the PCMCIA slots directly to the system bus and effectively passing through DMA signals if the system bus does have DMA capability;

c. a controlling circuit coupled to the PCMCIA slots, the determining logic circuit and to the internal system memory for controlling a DMA transfer if the system bus does not have DMA capability; and

d. a bus mastering circuit coupled to the controlling circuit, the determining logic circuit, the system bus and the CPU for disabling the CPU and mastering the system bus if the system bus does not have DMA capability.

13. The PCMCIA host adapter as claimed in claim 12 wherein there are two PCMCIA slots.

14. The PCMCIA host adapter as claimed in claim 13 wherein the means for controlling further comprises a first DMA controller circuit for controlling a first PCMCIA slot and a second DMA controller circuit for controlling a second PCMCIA slot.

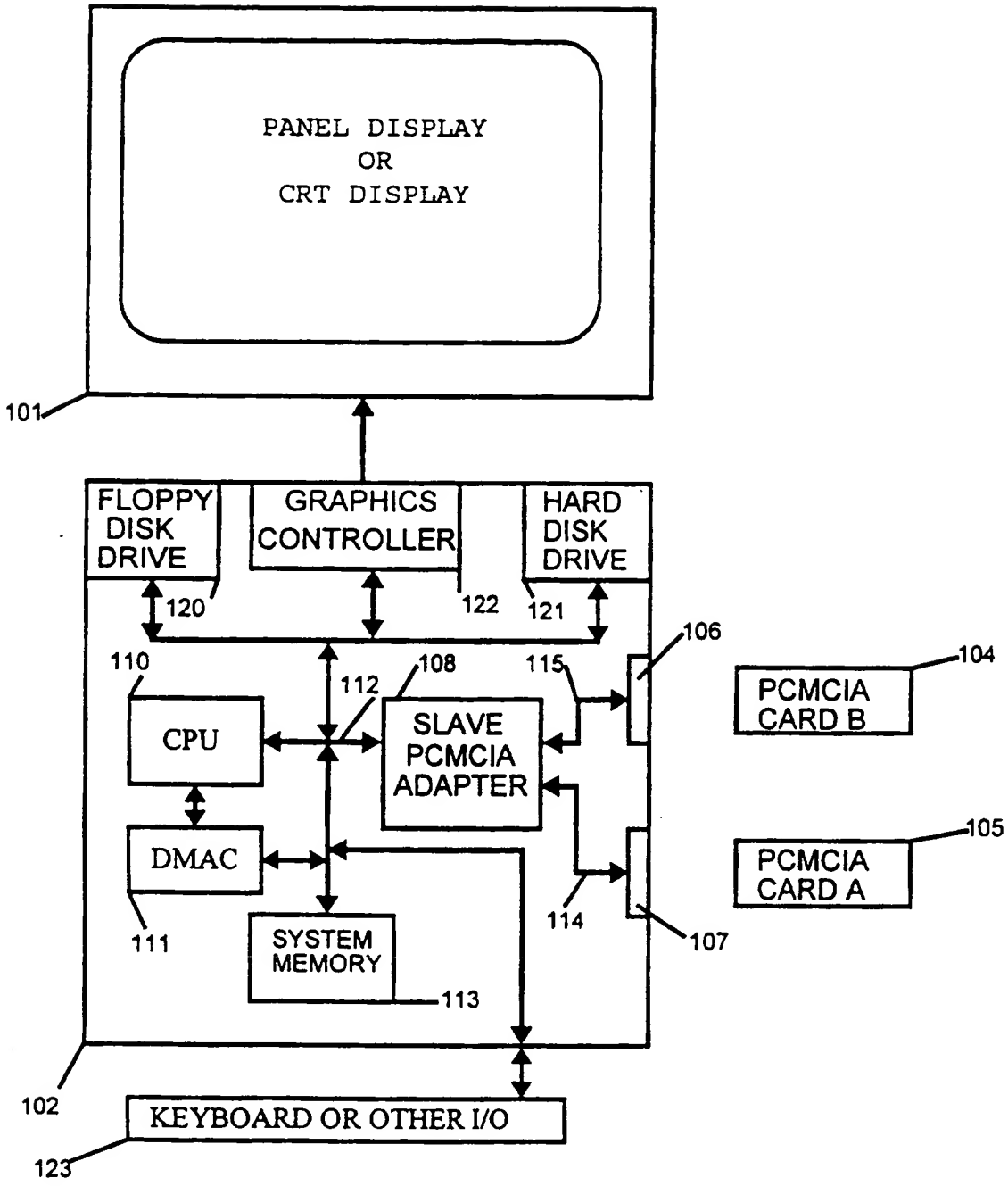


FIG. 1A
(PRIOR ART)

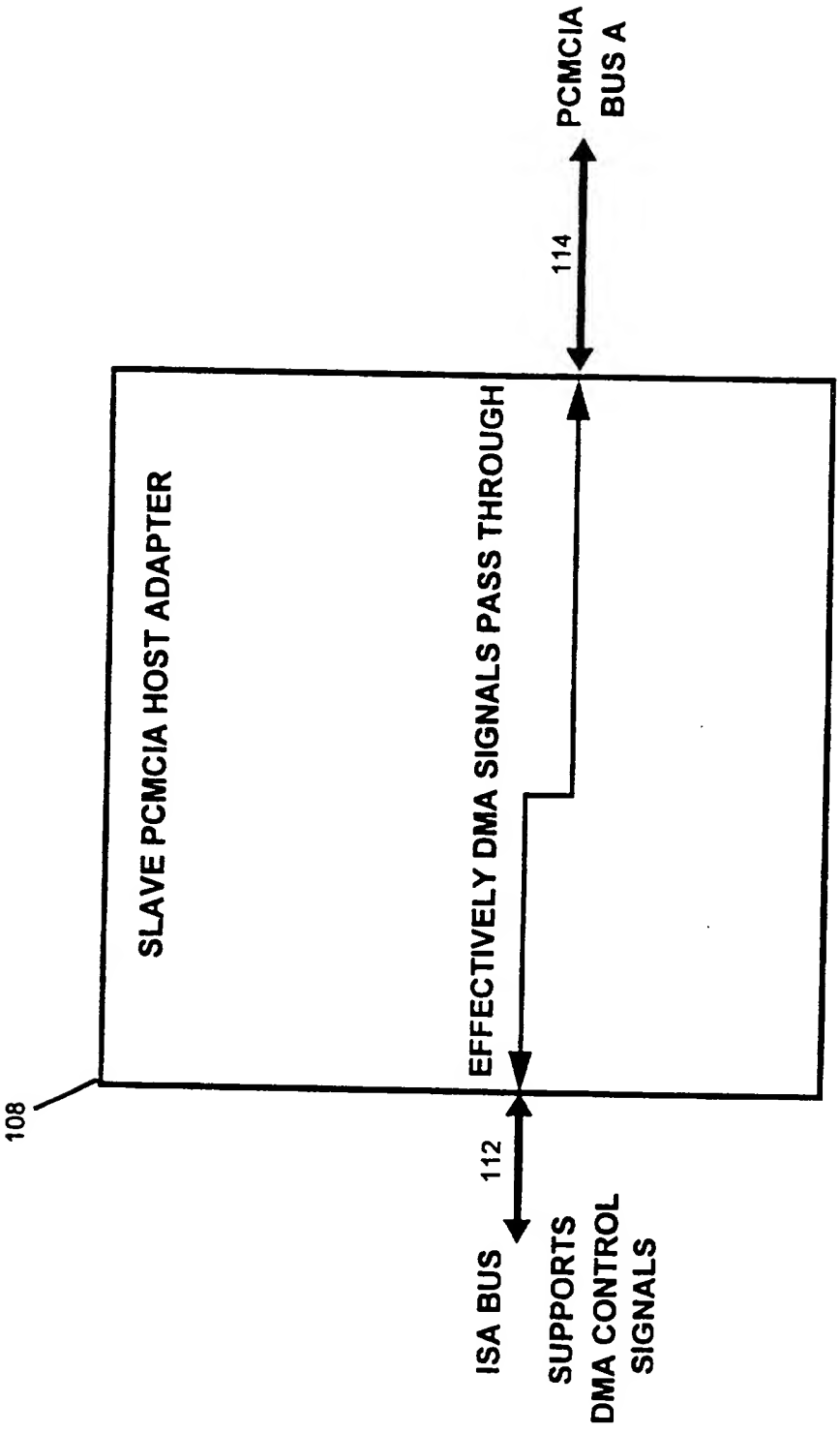


FIG. 1B
(PRIOR ART)

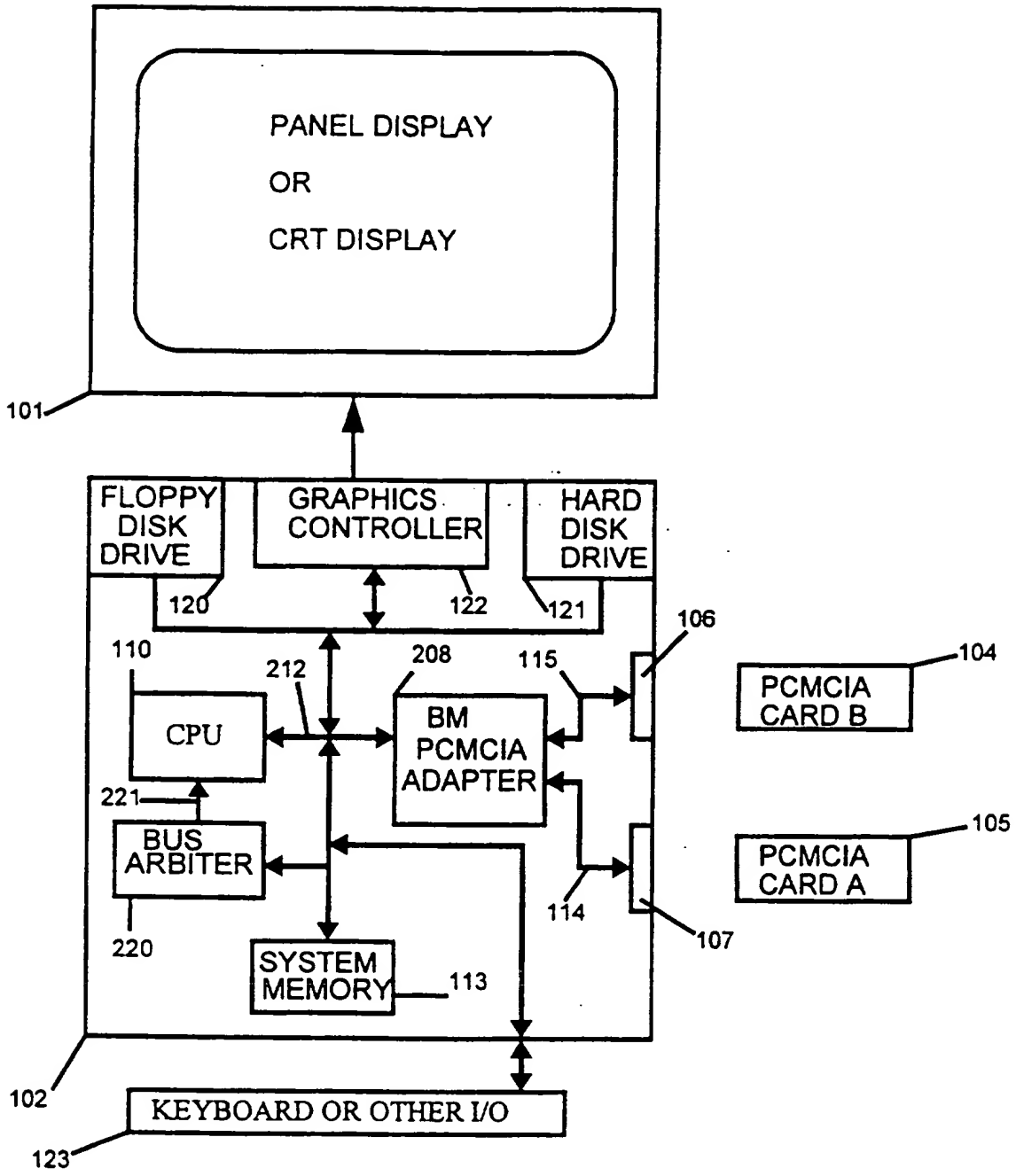

PANEL DISPLAY
OR
CRT DISPLAY
101
FLOPPY DISK DRIVE
GRAPHICS CONTROLLER
HARD DISK DRIVE
120
122
121
106
110
212
208
115
104
PCMCIA CARD B
CPU
BM PCMCIA ADAPTER
221
105
BUS ARBITER
PCMCIA CARD A
114
107
220
SYSTEM MEMORY
113
102
KEYBOARD OR OTHER I/O
123

FIG. 2

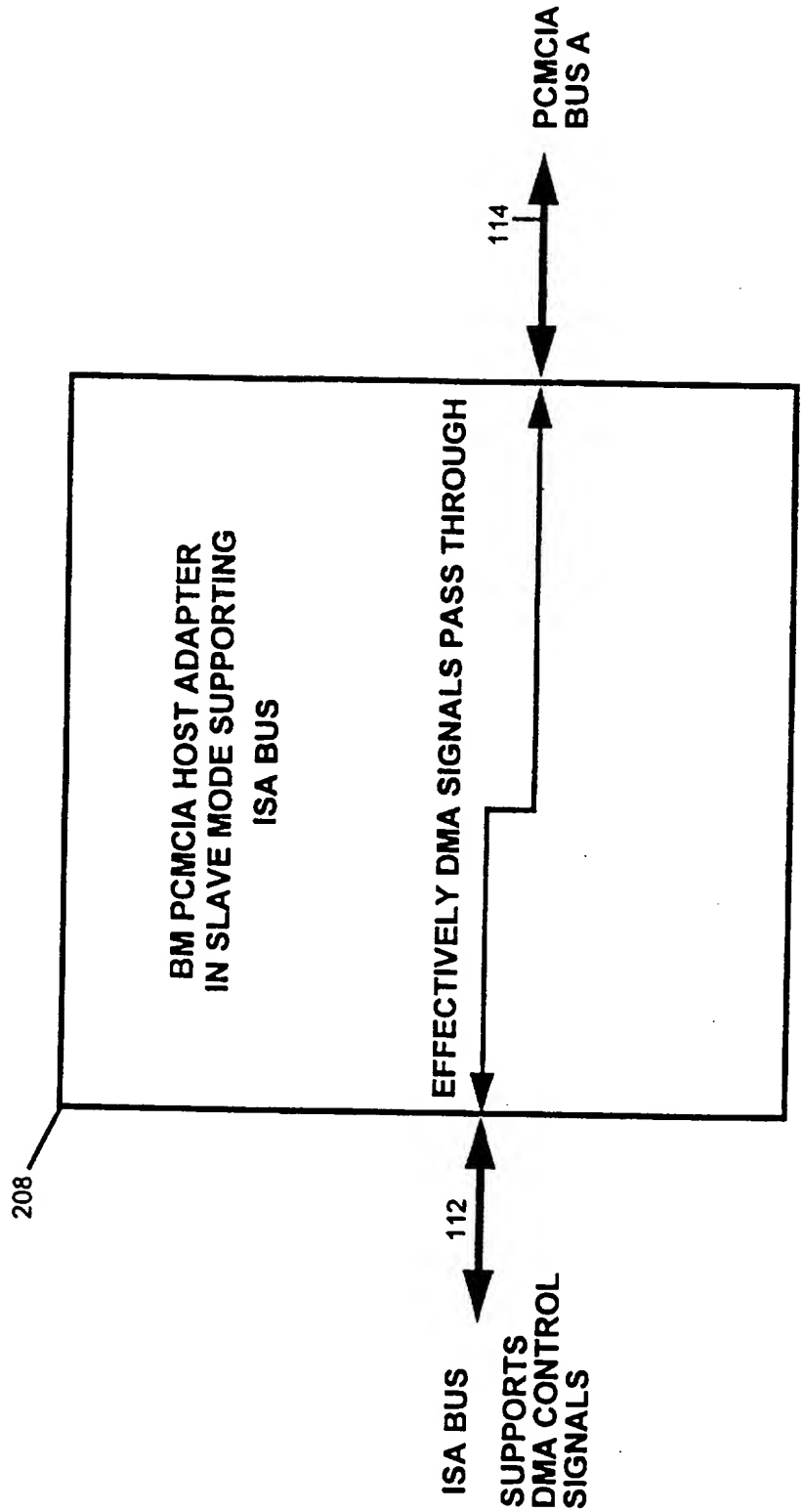

PCMCIA BUS A
114
BM PCMCIA HOST ADAPTER IN SLAVE MODE SUPPORTING ISA BUS
EFFECTIVELY DMA SIGNALS PASS THROUGH
208
112
ISA BUS SUPPORTS DMA CONTROL SIGNALS

FIG. 3

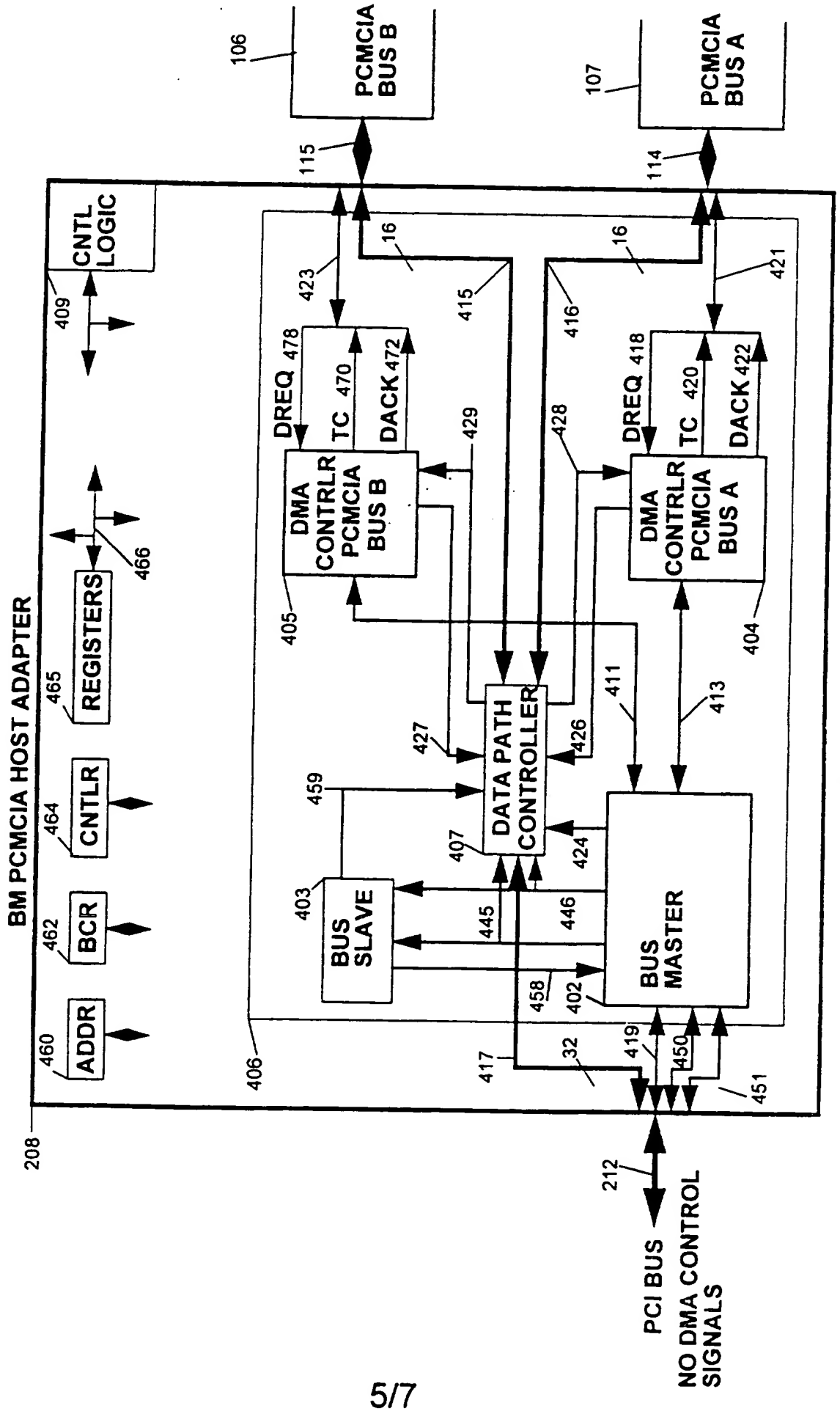

BM PCMCIA HOST ADAPTER
208
CNTL LOGIC
409
ADDR
460
BCR
462
CNTLR
464
REGISTERS
465
466
PCMCIA BUS B
106
PCMCIA BUS A
107
115
114
16
423
415
416
421
DREQ 478
TC 470
DACK 472
DREQ 418
TC 420
DACK 422
DMA CONTRLR PCMCIA BUS B
405
DMA CONTRLR PCMCIA BUS A
404
429
428
DATA PATH CONTROLLER
427
426
411
413
459
407
424
BUS SLAVE
403
BUS MASTER
402
445
446
458
406
417
32
419
450
451
212
PCI BUS
NO DMA CONTROL SIGNALS

FIG. 4

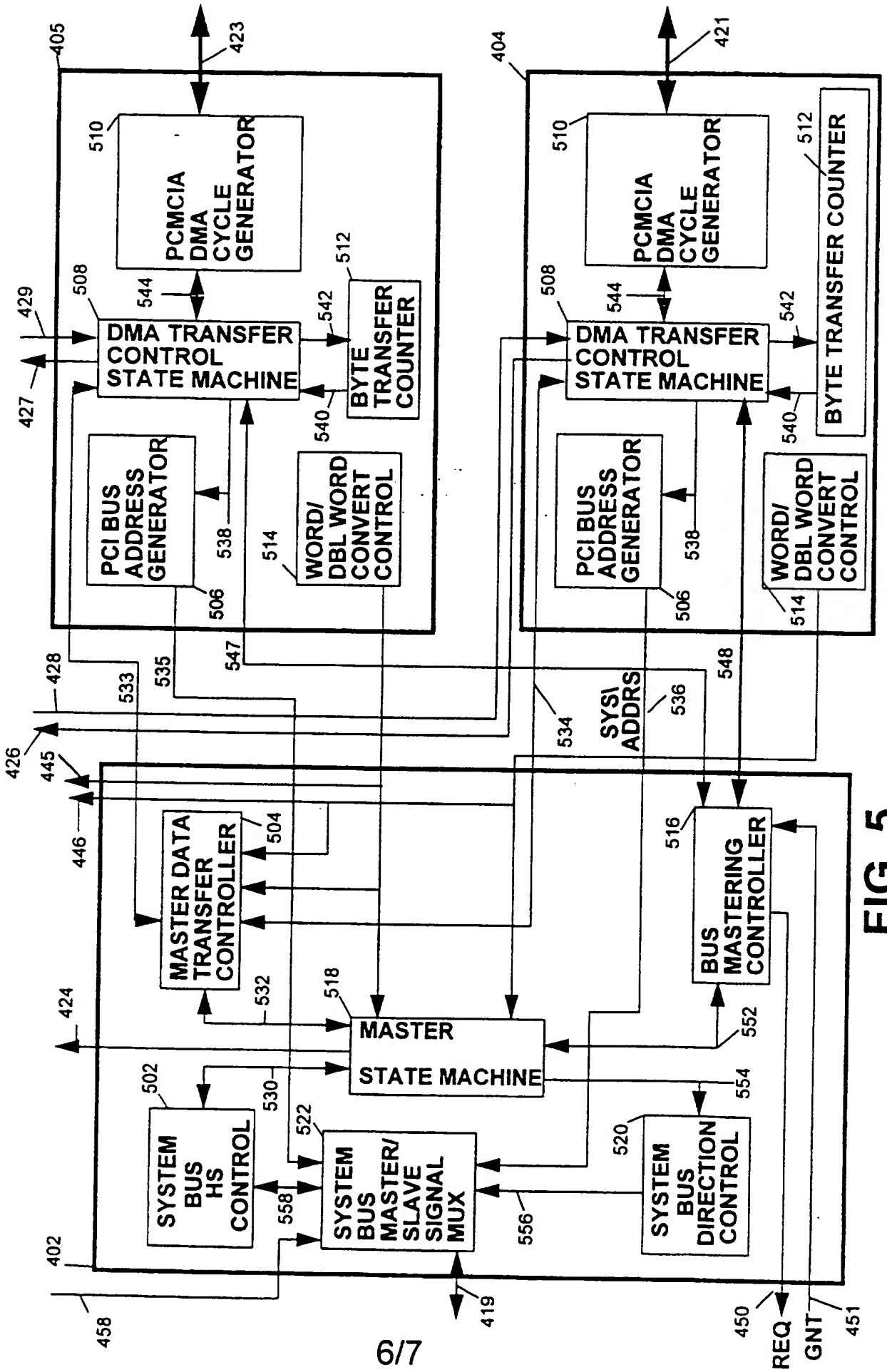

405
423
510
PCMCIA DMA CYCLE GENERATOR
512
BYTE TRANSFER COUNTER
544
508
DMA TRANSFER CONTROL STATE MACHINE
542
429
427
540
PCI BUS ADDRESS GENERATOR
538
514
WORD/ DBL WORD CONVERT CONTROL
506
404
421
510
PCMCIA DMA CYCLE GENERATOR
512
BYTE TRANSFER COUNTER
544
508
DMA TRANSFER CONTROL STATE MACHINE
542
540
PCI BUS ADDRESS GENERATOR
538
WORD/ DBL WORD CONVERT CONTROL
514
506
428
533
535
547
548
534
SYS\ ADDRS
536
426
445
446
MASTER DATA TRANSFER CONTROLLER
504
516
BUS MASTERING CONTROLLER
424
532
518
MASTER STATE MACHINE
552
554
502
530
522
SYSTEM BUS HS CONTROL
SYSTEM BUS MASTER/ SLAVE SIGNAL MUX
520
SYSTEM BUS DIRECTION CONTROL
558
556
402
458
419
450
REQ
GNT
451

FIG. 5

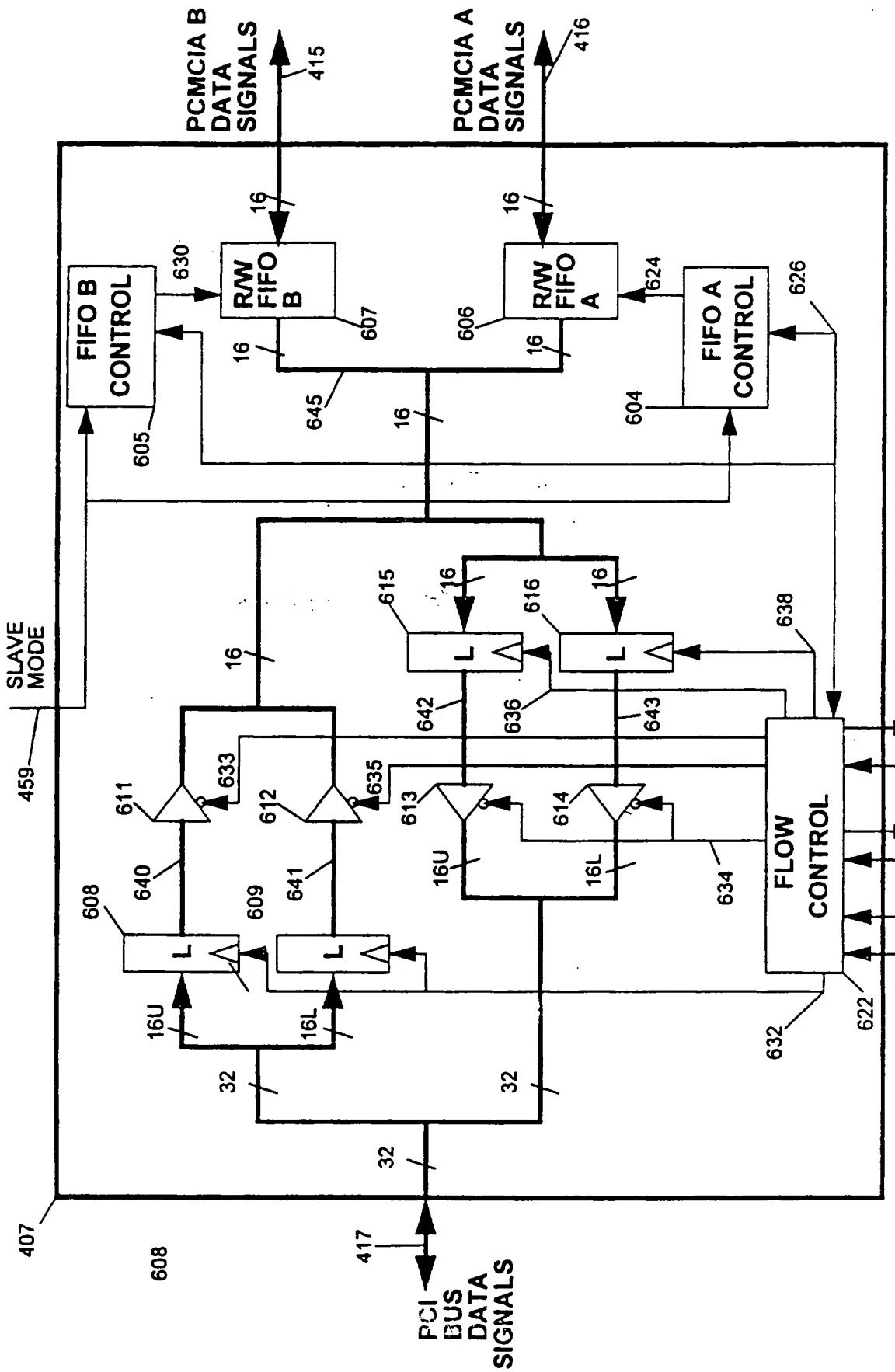

PCMCIA B DATA SIGNALS
415
PCMCIA A DATA SIGNALS
416
16
R/W FIFO B
607
R/W FIFO A
606
630
624
FIFO B CONTROL
605
FIFO A CONTROL
604
626
645
SLAVE MODE
459
615
616
638
L
642
636
643
633
635
611
612
613
614
640
641
609
608
16U
16L
634
FLOW CONTROL
429
427
428
426
446
424
622
632
32
407
417
PCI BUS DATA SIGNALS

FIG. 6